SW Engineering CSC648/848

Pawster

Dog meeting app

Section 01 Team 5

Milestone 4

May 4, 2022

Name:	**Role:**
Sina Pourdehmobed	Group Leader
John Paul Apolinar	Backend Lead
Jonathan Chen	Frontend Lead
Haozhan Li	Scrum Master
Umar Rama	Backend Lead
Henry Lam	Github Master

1. **Product Summary**

   Product name: Pawster

   - Any user should be able to Sign Up
   - Any user should be able to Login
   - Any user should be able to fill up own profile
   - Any user should be able to send a message to others user
   - Any user should be able to receive message by other user
   - Any user should be able to view listing users
   - Any user should be able to swipe card
   - Passwords should be encrypted
   - Any user should be able to use image link to upload image when they creating account
   - Matches you based on whether the other person swipes on you.

Product link: https://animaldatingapp-pawster-client.herokuapp.com/

Why Our Product is Unique

The point of our applications is to make the process of finding other pet owners who potentially would like to set up a playdate as fast as possible, with intuitive gestures and with as little hassle as possible. Many applications have too many steps and often, *too much* choice presented immediately to the user. This results in a lot of wasted time and potentially, a lot of time spent on simply *dealing* with the app rather than using it for its designed purpose. Pawster puts you on the quickest route from creating an account to talking with other users, allowing you to choose who and what you like, and simply unmatching (and reporting, if necessary) whenever you encounter a user that turns out to not be that great. Our app matches you based on whether the other person swipes right on you, meaning that both users have a vested interest in the other, and with our built in profile system, we allow users to put their best foot forward to truly be able to show who you are as a pet owner, and find others in your area that are interested in having their pets meet! Since our application saves all of your matches, you are able to re-open any past match (provided the two users are still matched) and message whenever and wherever you want! And thanks to our design being created specifically for ease of use and common gestures, it transfers to mobile quite well.

Client Webpage: https://animaldatingapp-pawster-client.herokuapp.com/
Server: https://animaldatingapp-pawster-server.herokuapp.com/

2. **QA Testing**

Unit Testing

Our approach to unit testing will be focused on testing how each component of our application works irrespective of the rest of the program. This means testing server calls, as well as front end features all independently of each other in order to ensure they all work as intended. At the current moment, we have several features implemented which require unit testing. The following features to unit test are:

- User sign up and account creation (front end)
- User sign up and account creation (back end)
- User "matching" feature
    - Display user specified potential matches (front end)
    - Fetch user specified potential matches (back end)
    - Display matched user results (front end)
    - "Swipe" feature to match with or skip a user (front end)
    - Store matched user results (back end)
- "Instant Messaging" feature

- Display past message history (front end)
- Store message history between users (back end)
- Show past matched users (front end)
- Store past matched users (back end)

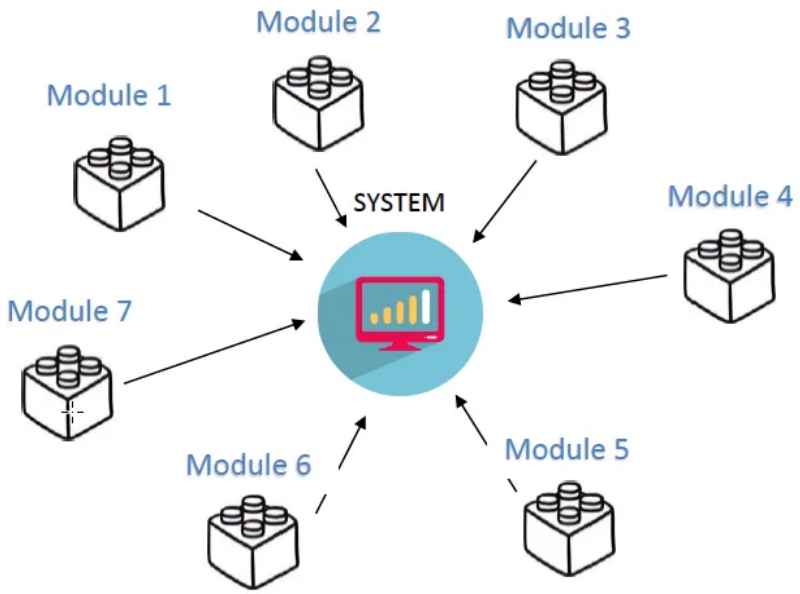
Integration Test

There is a simple algorithm a QA team can apply to run integration tests. Here is what the sequence of our actions typically looks like:

- Write a test plan.
- Create test cases and use cases.
- Run tests after unit integration.
- Detecting errors.
- Retest the functionality after bug fixing.
- Repeat the testing cycle until all bugs are fixed.

Integration testing is a type of testing meant to check the combinations of different units, their interactions, the way subsystems unite into one common system, and code compliance with the requirements. Our approach to check login and sign up features in this app, we view them as separate units. We check the ability to log in and sign up after a user creates an account or logs in. For integration testing, our team will be using components that have already been tested as separate units. We group these units into sets and check them in accordance with the test plan. Unit testing is the initial stage of the QA process. It prepares the functionality to the following stage which is integration testing. Without passing the former and verifying that units perform correctly, we cannot proceed to the latter and start putting them together.

For example like each module is tested so that the desired outcome of the system is met:



Integration testing and unit testing are two levels of software testing, where a unit is the basic one and integration is the sequential one. Unit testing implies checking the smallest functioning parts of code separately. Integration testing goes further: we look for errors that happen when units start interacting. The fact that units have successfully passed the tests at the previous stages doesn't guarantee that they will work well together.

Integration testing allows dividing code into blocks that consist of several units and checking the parts of software gradually before assembling them all into a complete system. It means that all the systems will be properly revised, regardless of when, how, and why they have been created.

Verification of software integrity is essential since only functional and highly-performing builds should be allowed to proceed to the further stages of testing. Otherwise, finding bugs will become more difficult and take more time.
Advantages of the integration testing are:

- Simultaneous testing of different modules is very convenient, practical, and cost-efficient.
- Integration checks can take place at any stage of the SDLC.
- It is very convenient for a team involved in projects with constantly changing requirements or logic that is under review after development has started.
- Unlike other types of tests, integration can cover any amount of program code in one sprint.

3. **Code Review**

In order to effectively relay our work without any conflicting technical or syntactical issues, our team has collectively determined that we should all work on individual segments of specific functionalities in relation to user interface as well as back end functionalities. We were able to enforce our strategy effectively by referring through our team member's previous changes/requests to inspect any potential issues or inconsistencies prior to working on their respective parts. In order to accomplish this, we used a workflow that included implementing "feature" branches, in which two team members were assigned a feature in which to implement, where the work could be split evenly.

Each feature was specific to each group of two, where they implemented their feature and commented it out, letting the team ahead of them know what feature was implemented, and what new functions were available to use. At the end of each feature implementation, a final pull request was created and reviewed with the team, as well as the GitHub master, where any potential merge conflicts and errors of any sort were ironed out, as well as any clarifications on certain functioning parts were made at this time. We make sure there are no merge conflicts regarding the pull requests. With six people in our team, the goal was to implement at least three

features a week, assigning one feature to each pair of people, which we decided would work well for a weekly sprint, organized by our scrum master. Through thorough communication on Discord, we were able to effectively communicate and complete this milestone.

The following are Pull Requests made in relation to P1 features. Commentaries in regards to each respective pull request were thoroughly reviewed and discussed via discord prior to committing the changes.

1) Home.js https://github.com/CSC-648-SFSU/CSC648-spring22-01-Team05/pull/11
2) OnBoarding.js https://github.com/CSC-648-SFSU/CSC648-spring22-01-Team05/pull/12
3) Nav.js https://github.com/CSC-648-SFSU/CSC648-spring22-01-Team05/pull/14
4) index.js https://github.com/CSC-648-SFSU/CSC648-spring22-01-Team05/pull/16
5) dashboard.js https://github.com/CSC-648-SFSU/CSC648-spring22-01-Team05/pull/17
6) Login: https://github.com/CSC-648-SFSU/CSC648-spring22-01-Team05/pull/21
7) Login #2: https://github.com/CSC-648-SFSU/CSC648-spring22-01-Team05/pull/19
8) Theme: https://github.com/CSC-648-SFSU/CSC648-spring22-01-Team05/pull/15
9) Storing information for users in cookies: https://github.com/CSC-648-SFSU/CSC648-spring22-01-Team05/pull/18

4. Self-check: Adherence to original Non-functional specs

Initial Non-Functional Requirements	
Requirements:	**Status:**
Data will be stored using mongodb and the application will be hosted using Amazon AWS.	DONE
The application should be compatible with different types of browsers.	DONE
Code will be in the master branch of github and working at all times.	DONE
Application should be working for the duration of the project.	ON TRACK
Website should be easy to use, users can find things easily from the top menu	ON TRACK